UNIVERSIDADE FEDERAL SÃO JOÃO DEL REI - UFSJ

CHRISTOFFER DE PAULA OLIVEIRA

JOÃO AUGUSTO DA CRUZ

DOCUMENTAÇÃO

SÃO JOÃO DEL -REI, 29/11/2015

UNIVERSIDADE FEDERAL DE SAÕ JOÃO DEL REI - UFSJ

TRABALHO PRÁTICO

CHRISTOFFER DE PAULA OLIVEIRA

JOÃO AUGUSTO DA CRUZ

Trabalho apresentado como parte dos requisitos necessários para aprovação na disciplina Algoritmo e Estrutura de Dados do curso Ciência da Computação da Universidade Federal de São João Del Rei – UFSJ, desenvolvida sob a orientação do (a) Prof. (a) Carolina Ribeiro Xavier

São João Del – Rei,29/11/2015

Sumário

1 INTRODUÇÃO

Documentação da implementação do trabalho pratico de AEDS.

Nesta implementação tem se como objetivo virtualizar um jogo da velha com opções de dois tipos de jogos, JOGADOR vs COMPUTADOR e JOGADOR vs JOGADOR, no primeiro item citado há uma estrutura de dados com a intenção de fazer com que o próprio computador faça suas próprias jogadas, já no segundo item os dois jogadores serão responsáveis por suas próprias jogadas. O algoritmo está com instruções para o que o usuário saiba o que deverá ser digitado e está também com validações para que não seja passado ao programa números fora do parâmetro.

As próximas páginas deste documento trata-se de um aprofundamento maior sobre os itens citados, diversos desafios(problemas) enfrentados seguidos de suas respectivas soluções e também justificando as estruturas usadas.

2 IMPLEMENTAÇÃO

2.1 SOBRE A IMPLEMENTAÇÃO

O jogo se baseia no recebimento das coordenadas digitadas pelo usuário, preenchendo o lugar corresponde a elas, após isto as coordenadas são adaptadas para a matriz [6][6],pois o usuário apenas pode identificar uma matriz[3][3],depois dessas etapas e verificado se o jogador ganhou em algumas das posições possíveis para encerrar o jogo atual e imprimir quem venceu.Demonstração:

if((contdrod<9 && vencedor==0))//condição para que jogador 2 não faça jogadas caso há um vencedor ou tenha dado velha.

```
{
    do//validando entrada do caracter no jogo
    {
        do//validadando as coordenadas do usuário 2.
        {
        printf("Vez do %s jogar\ndigite as coordenadas que deseja marcar(linha coluna) e que nao esteja marcada\n",player2);
        scanf("%d",&coo1);
        scanf("%d",&coo2);
```

```
}while((coo1>3) || (coo1<0) || (coo2>3) || (coo2<0));//condição para que as coordenadas digitadas seja válidas

if(coo1==3)//Deixando o caracter digitado de acordo com a matriz[6][6].

{

        coo1=5;

}

if(coo1==2)//Deixando o caracter digitado de acordo com a matriz[6][6].

{

        coo1=3;

}

if(coo2==3)//Deixando o caracter digitado de acordo com a matriz[6][6].

{

        coo2=5;

}

if(coo2==2)//Deixando o caracter digitado de acordo com a matriz[6][6].

{

        coo2=3;

}

}while(matriz[coo1][coo2]==crt1 || matriz[coo1][coo2]==crt2);//condição para que o jogador não marque onde já está marcado.

contdrod++;

vezjogador=1;//comando para alternar entre a vez de cada jogador jogar.

matriz[coo1][coo2]=crt2;//colocando  o caracer no jogo

if((matriz[1][1]==crt2 && matriz[1][1]==matriz[3][1] && matriz[1][1]==matriz[5][1])

|| (matriz[1][1]==crt2 && matriz[1][1]==matriz[3][3] && matriz[1][1]==matriz[5][5])
```

```
                || (matriz[1][1]==crt2 && matriz[1][1]==matriz[1][3] &&
                matriz[1][1]==matriz[1][5])

                || (matriz[1][3]==crt2 && matriz[1][3]==matriz[3][3] &&
                matriz[3][3]==matriz[5][3])

                || (matriz[1][5]==crt2 && matriz[1][5]==matriz[3][5] &&
                matriz[3][5]==matriz[5][5])

                || (matriz[1][5]==crt2 && matriz[1][5]==matriz[3][3] &&
                matriz[3][3]==matriz[5][1])

                || (matriz[3][1]==crt2 && matriz[3][1]==matriz[3][3] &&
                matriz[3][3]==matriz[3][5])

                || (matriz[5][1]==crt2 && matriz[5][1]==matriz[5][3] &&
                matriz[5][3]==matriz[5][5]))//Verificando se o jogador 2 ganhou.

                {

system("clear");

printf("\t!!!!!JOGADOR %s VENCEUUU!!!!!\n",player2);

vencedor=1;

sit_game(matriz,6);

getch();

}

}
```

No algoritmo , as validações não impedem, quando for pedido números, que o usuário coloque outros caracteres da tabela ASCIII , como por exemplo ☼,► e◄, mas impede de colocar números fora do parâmetro. No tipo de jogo usuário versus computador, mesmo removendo algumas táticas do usuário, ele assim como o computador tem chance de vencer, lembrando que é sempre o usuário que inicia no jogo, assim dado uma certa quantidades de jogadas o computador passa a jogar com base no que o usuário jogou. E como não e randômico se o usuário ganhar do computador uma vez, ele sempre poderá ganhar novamente da mesma forma que ganhou inicialmente.

2.2 VARIÁVEIS

Abaixo estão listadas a utilidade de cada variável utilizada na implementação

int opc-> variável responsável por guardar a opção escolhida pelo usuário. Sendo as opções jogador vs computador, jogador vs jogador e encerrar o programa.

int vezjogador->variável para controlar a vez de cada jogador sendo inicializada com 1 para que o jogador 1 seja o primeiro a jogar.

int coo1->variável para armazenar a coordenada(linha) digitada pelo usuário.

int coo2->variável para armazenar a coordenada(coluna) digitada pelo usuário.

int contdrod->variável para contar quantas jogadas foram feitas, auxiliando ainda no processo de quando o jogo dará velha.

int vencedor->variável responsável por controlar quando há vencedor no jogo auxiliando ainda nas condições para encerrar o jogo se houver vencedor e para não deixar que sejam feitas jogadas após houver jogador.

int ctrljogadacpu->variável para contar se o computador fez uma jogada, auxiliando ainda nas condições responsáveis por controlar o computado, para que não seja feita duas jogadas pelo mesmo.

char player1[30]->vetor para armazenar o nome do jogador 1.

char player2[30]->vetor para armazenar o nome do jogador 2.

char crt1->variável por armazenar o carácter do jogador 1.

char crt2->variável por armazenar o carácter do jogador 2.

char matriz[6][6]={' ','1',' ','2',' ','3','1',' ','|',' ','|',' ',' ',' ','-','-','-','-','-','-','2',' ','|',' ','|',' ',' ',' ','-','-','-','-','-','-','3',' ','|',' ','|',' '}; -> Matriz onde coloca as coordenadas e as linhas que montam o jogo da velha.

2.3FUNÇÕES

Foram utilizadas neste algoritmo apenas duas funções são elas:

void sit_game(char matriz[][6],int n) ->Função responsável por mostrar(imprimir) a situação do jogo(matriz) naquele momento .

void reset_game(char matriz[0][6],int n,char crt1,char crt2) ->Função que após o término do jogo reinicia a matriz colocando espaço nos lugares dos caracteres colocados pelos usuários e computador.

2.4 PROBLEMAS

A princípio uma dificuldade que surgiu foi como imprimir a estrutura do jogo da velha que além da estrutura normal(cruzes), teria ainda as coordenadas para ajudar o usuário a ver onde quais números representava as casas.

Outras duas dificuldades mas não muito grande foi encontrar onde no jogo da velha poderia ganhar e alternar a jogada entre os usuários.

E por último um problema enfrentado foi fazer com o que o computador jogasse nas posições adequadas.

2.5 SOLUÇÕES

Para preencher a matriz com cruzes e coordenadas foi criada uma matriz 6x6, onde é armazenado os traços que formam a cruz, as coordenadas e os espaços para serem preenchidos.

char matriz[6][6]={' ','1',' ','2',' ','3','1',' ','|',' ','|',' ',' ','-','-','-','-','-','2',' ','|',' ','|',' ',' ','-','-','-','-','-','3',' ','|',' ','|',' '};//preenchendo a matrix 6x6

Assim para imprimir na tela é possível usar formas simples de impressão utilizadas para qualquer matriz 6x6

Para saber onde é possível ganhar no jogo da velha foi feita uma análise no jogo e após esta análise é possível perceber que há oito linhas que é possível ganhar e em cada linha deve-se ter 3 caracteres iguais ,assim 8x3 momentos que se pode ter a vitória .

2.5.1 JOGADOR vs COMPUTADOR

Ainda que nesta parte a implementação tenha as estruturas básicas citadas no inicio do item 2.5 ela se diferencia pelo fato de que as condições para o computador jogar estão todas nela esta parte está no ‘case 1’ da implementação.

Um problema enfrentado foi fazer com o que o computador jogasse nas posições adequadas para isso foi usado uma lógica com base na análise feita citada no último parágrafo do item 2.5. Foi criado uma estrutura de lógica para que o computador

verificasse em primeiro lugar se ele poderia ganhar, em segundo lugar se na próxima jogada o jogador tem chance de ganhar e por ultimo se o ele não tiver feito nenhuma jogada, ele entra em uma condição em que há jogadas estratégicas para remover algumas táticas do usuário e para tentar ganhar. Esta estrutura lógica se compõe basicamente de ifs para verificar cada local do jogo da velha que pode ser completado nos dois primeiros instantes ,ou seja, quando o computador pode ganhar e quando o computador deverá bloquear o usuário para que ele não vença, depois disso a mais um conjunto de ifs com a finalidade de bloquear certas táticas do usuário ,como exemplo colocar nas laterais. Apenas um exemplo do comando if utilizado para verificar onde o computador devera marcar:

```
if((matriz[1][1]=='#') && (matriz[1][3]=='#') && (matriz[1][5]!=crt1))

//Verificando se o computador tem chance de ganhar nas posições 1x1 e 1x3 e somente se não estiver caractere do usuário ele passa da condição.

{

        matriz[1][5]='#';     //Então é marcada a posição da vitória.

        ctrljogadacpu++;   //Assim o a variável controladora de jogadas recebe mais 1 que junto com mais condições não oermite que o computador faça outra jogada.

}
```

Para bloquear o usuário ao invés de verificar se o computador tem chance de ganhar é verificado se o jogador tem chance de vencer trocando o ‘#’ por ‘crt1’, e para não remarcar no local já marcado e desperdiçar uma jogada foi apenas trocado o ‘crt1’ por ‘#’ no final da condição.

2.5.2 JOGADOR vs JOGADOR

Está parte da implementação está no “case 2:” onde o usuário pode jogar com outro usuário escolhendo os carácteres a serem utilizados e também seus nomes de jogadores. Basicamente funciona como citado no início do item 2.1 e uma parte que diferencia é a estrutura de código usada para alternar entre usuários.

Para alternar entre os usuários foi utilizada a variável ‘vencedor’ e a cada jogada de um usuário esta variável recebe um valor ,1 ou 2, com as condições estabelecida e a cada loop, uma condição diferente é verdadeira e cada condição representa um usuário, assim cada usuário jogará em cada loop.

```
if(vezjogador==1) //condição para alternar a vez dos jogadores. observe que o jogador um jogou

{
```

vezjogador==2;

}

if(vezjogador==2)//condição para alternar a vez dos jogadores.obeserve agora que esta condição se tornou verdadeira no loop seguinte.Sendo a vez do jogador 2

{

vezjogador==1; //Agora no próximo loop a condição para qu o jogador 1 faça a jogada será verdadeira.

}

Lembrando que está parte é apenas um exemplo simplificado do que realmente tem na implementação.

3 CONCLUSÃO

Enfim podemos concluir que o funcionamento do programa garante que seja virtualizado um jogo da velha que é jogado com coordenadas, semelhante a batalha naval, e que reconhece quem é o vencedor ou se deu velha, chamando os usuário pelo nome cadastrado, não permitindo caracteres iguais entre usuários e entre usuário e computador e também não permitindo espaço, tudo isso para melhorar a experiência com o jogo.